42.P17498 Patent Application

APPLICATION FOR UNITED STATES LETTERS PATENT

For

MOTION ESTIMATION SUM OF ALL DIFFERENCES (SAD) ARRAY HAVING REDUCED SEMICONDUCTOR DIE AREA CONSUMPTION

Inventors:

Louis Lippincott

Kalpesh Mehta

Prepared by:

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard
Los Angeles, CA 90025-1026
(408) 720-8300

MOTION ESTIMATION SUM OF ALL DIFFERENCES (SAD) ARRAY HAVING REDUCED SEMICONDUCTOR DIE AREA CONSUMPTION

Field of Invention

[0001] The field of invention relates to motion estimation; and, more specifically, to a motion estimation Sum of All Differences (SAD) logic circuit having reduced semiconductor die area consumption.

Background

[0002] Digital video is typically embodied as a collection of digital still frames that convey motion when displayed in succession. Because visual information (such as digital video) naturally consumes large amounts of data, data compression is often applied to reduce the amount of data without substantially degrading the original imagery. Motion estimation is a compression technique that is most often applied to digital video.

[0003] According to the motion estimation approach, a first frame is broken down into small pieces (referred to as "macro blocks"); then, each macro block is searched for in a data field (referred to as a "search window") of a second frame that follows or precedes the first frame (in the sequence of frames that the video information is comprised of). The macro block being searched for may be referred to as a "reference macro block" or a "reference block". For each found reference macro block (ideally all reference macro blocks are found in the second frame), a displacement vector is recorded that describes the movement

of the reference macro block. Thus, compression is achieved by preserving displacement vectors instead of actual visual content.

[0004] **Figure 1** demonstrates a simple example of motion estimation for a digital video that depicts an automobile moving from left to right. A "current" frame 103 corresponds to the first frame referred to above. The current frame 103 follows a "previous" frame 101 in the sequence of frames that convey movement of the automobile from left to right in **Figure 1**. The previous frame 101 corresponds to the second frame referred to just above. A reference macro block 102b that captures the nose of the automobile is depicted in the current frame 103 and a search window 104 is depicted in the previous frame 101.

[0005] It is important to point out that other scenarios are possible as between which frame is the "first" frame having the reference block and which frame is the "second" frame having the search window. For example, alternatively, the second frame may be a subsequent frame relative to the first frame (rather than a previous frame as described just above with respect to **Figure 1**).

[0006] Note that other reference macro blocks would be defined in current frame 103. Notably, the sequence of the previous and current frames 101, 103 portray the automobile moving a distance K to the right. According to basic motion estimation, a field of data 104 larger than a macro block is taken from the previous frame 101 and used as a search window for finding the reference macro block 102b of the current frame 103. The search window 104 may be centered around the center position of the current frame's reference macro block 102b (i.e., the search window 104 is centered at a horizontal offset of K from the frame

origin in the example of **Figure 1**). An aptly size search window 104 should always result in the finding of the reference macro block 102b (if the object undergoing movement does not change its own features). According to the example of **Figure 1**, the reference macro block 102b of the current frame 103 is discovered in the search window 104 a distance K to the left of its original position within the current frame 101. As such, a displacement vector of (K,0) is recorded for macro block 102a (i.e., $((x_c-x_p),(y_c-y_p)) = ((K-0),(0-0) = (K,0)$; where, "c" denotes "current" and "p" denotes previous).

Figures

[0007] The present invention is illustrated by way of example, and not limitation, in the figures of the accompanying drawings in which like references indicate similar elements and in which:

[0008] **Figure 1** shows an exemplary depiction of two digital video frames;

[0009] **Figure 2** shows an exemplary depiction of SAD logic circuitry and its corresponding macro block data and search window data;

[0010] **Figure 3** shows an exemplary depiction of an AD unit array coupled to receive a previous frame's data and current frame's data;

[0011] **Figure 4a** shows an exemplary depiction of an AD unit having improved power and surface area consumption and its corresponding macro block data and search window data;

[0012] **Figure 4b** relates to the different resolutions between a SAD calculation process that processes all bits per data value and an SAD calculation process that processes less than all bits per data value;

[0013] **Figure 5a** shows an exemplary depiction of another AD unit embodiment having improved power and semiconductor die area consumption and its corresponding macro block data and search window data;

[0014] **Figure 5b** relates to the different resolutions between a SAD calculation process that processes all bits per data value and a SAD calculation process that can be performed with the circuitry depicted in Figure 5a;

[0015] **Figure 5c** shows a methodology for determining which M bits to use for SAD calculation purposes when video data values are naturally expressed as N values;

[0016] **Figure 6** shows a circuit that can be designed into a system that employs video compression.

Detailed Description

[0017] In order to "find" a macro block in a search window, a Sum of Absolute Differences (SAD) approach is often used. **Figure 2** shows a depiction of a basic architecture. According to the basic architecture of **Figure 2**, SAD logic circuitry 201 calculates the absolute value of the difference (i.e., hereinafter, "the absolute difference") between a data value (e.g., a pixel value) from the first frame's reference macro block data 202 and its corresponding data value (e.g., a corresponding pixel value) from the second frame's search window data 203. Here, use of the term corresponding means that data values chosen for calculating absolute difference terms are aligned with respect to their positioning within the first frame's reference macro block and a reference macro block worth of data within the search window data 203.

[0018] That is, basic operational flow involves calculating and summing over absolute differences for each pair of corresponding data values between the first frame's reference macro block data 202 and a specific macro block worth of data within the search window data 203. Once the sum of the absolute differences for the specific macro block worth of data is determined, the basic operational flow is repeated for a next specific macro block worth of data within the search window data 203. When a sum of absolute differences has been determined for all macro blocks worth of data that can be identified within the search window data 203, the macro block worth of data having the lowest sum of absolute differences is deemed the "found" macro block within the search window.

[0019] For example, **Figure 2** shows a "lower right corner" value 204 in the reference macro block data 202. For purposes of calculating an absolute difference, the lower right corner value 204 is always compared against its corresponding value (i.e., the lower right corner value) of each macro block worth of data that can be identified within the search window 203. For simplicity, **Figure 2** only shows two macro blocks worth of data 205, 206 (amongst the many) that can be identified within the search window 203. Calculation of the SAD for macro block 205 will therefore include calculating the absolute difference of the corresponding lower right corner values 204 and 207. Likewise, calculation of the SAD for macro block 206 will include calculating the absolute difference of the corresponding lower right corner values 204 and 208. All corresponding values between the reference macro block and the search window macro blocks are similarly dealt with.

[0020] The depiction of **Figure 2** and its corresponding discussion above related to the processing involved for only a single reference macro block. Here, it is important to recall that the first frame is broken down into a plurality of many reference macro blocks. As such, the processing referred to above with respect to **Figure 2** is performed many times over for each reference macro block within the first frame so that compression can be achieved for the area of an entire frame. In order to achieve high performance (e.g., reduced processing times) an array of Absolute Difference (AD) units is used to simultaneously calculate AD terms for a plurality of corresponding data value (e.g., pixel) pairs as between a reference macro block and a macro block worth of data from a search window.

[0021] **Figure 3** explores the concept in more detail. **Figure 3** shows SAD logic circuitry 301 comprised of an array 314 of AD units 11 through XY each geared to calculate the absolute difference between data from a corresponding location within the reference macro block and a macro block worth of data within a search window. For example, for macro block A 305, AD unit 33 is designed to calculate the absolute difference between: 1) a data value 311 that is held within a register 302 that stores the reference macro block's data; and, 2) a corresponding data value 312 to data value 311 within macro block A 305 which, in turn, resides within a search window's worth of data that is held within a random access memory (RAM) 303.

[0022] For the same macro block A 305, SAD unit (X-1)(Y-1) is designed to calculate the absolute difference between: 1) a data value 304 that is held within a register 302; and 2) a corresponding data value 307 to data value 304 within macro block A 305. Likewise, for macro block B 306, AD unit 33 is designed to calculate the absolute difference between data value 311 and data value 313; and, AD unit (X-1)(Y-1) is designed to calculate the absolute difference between data value 304 and data value 308.

[0023] The absolute differences for all the data values are added to together by an adder 315 to form a SAD value for the particular macro block within the search window being analyzed (e.g., macro block_A or macro block_B). The computed SAD value is then compared by comparator 316 against the lowest SAD value obtained so far within the search window (which is stored in register 317). When the search window has been completely analyzed, the macro block

within the search window having the lowest SAD value is recognized as the "found" reference block within the search window. A motion vector is then calculated. With each new reference macro block loaded into register 302, a new field of search window data is loaded into RAM 303.

[0024] A problem, however, is that large arrays of AD units consume large amounts of semiconductor die area and electrical power.

[0025] In order to address the power and semiconductor die area consumption issues, it is helpful to recognize that a SAD calculation acts to compare likeness between two fields of data (e.g., current frame's macro block and previous frame's search window macro blocks); while, the specific fields of data themselves contain data points that express different intensity values. As such, efficiency can be realized by recognizing that the number of bits needed per data value (e.g., per pixel) to allow a range of intensity values for images of sufficient contrast (e.g., typically 8 bits per pixel) is greater than the number of bits needed per data value to confirm or deny sameness between fields of data.

[0026] Simply put, the resolution of the data (in terms of the number of bits per data value) does not need to be as high for SAD calculations as it does for image rendering. An AD unit within an AD array can therefore be designed to process less than the full number of bits per data value than what is provided by the frames themselves; and, as a consequence, conserve power and semiconductor die consumption.

[0027] **Figure 4a** shows an example. Note that **Figure 4a** indicates that data values from the reference macro block 402 and the macro block being analyzed

within the search window 209 naturally take the form of N bit wide data values (i.e., N bits per data value). Yet, the (x^{th}, y^{th}) AD unit 418 only processes M of the N bits per data value, where, M is less than N. By processing less than all of the bits per data value from the frame data, the AD unit 418 should be smaller in size and consume less power than an AD unit configured to process full width data values. Such efficiencies should extend to the array 414 as a whole because it is comprised of a plurality of efficient AD units.

[0028] **Figure 4a** suggests that the reduction in bit width from N to M bits is accomplished by reading a full N wide data value from both the reference macro block 402 (e.g., from register 302 in **Figure 3**) and the macro block within the search window 409 (e.g., from RAM 303 in **Figure 3**); and, storing the N wide data values into respective register space 410, 411. Then, only the M bits used for SAD calculations are read from the respective register space 410, 411 by the AD unit 418. In other implementations, the loaded frame data itself may be properly reduced beforehand from N bits per value to M bits per value (e.g., prior to being loaded into register 302 and RAM 303, respectively); or, the loaded frame data may remain as N wide data values but the wiring is configured to ignore the unused N-M bits when the data is physically transferred toward the AD unit 418. Either of these basic architectures could be represented by removing register space 410 and 411 from Figure 4a and connecting the reference block data 402 and the search window data 404 to the AD unit with an M wide bus (i.e., the N wide bus is removed).

[0029] With the understanding that efficiency is gained through SAD calculations involving less bits per data value than the image data itself, the question arises as to which bits should be ignored and which bits should be selected for purposes of performing SAD calculations. The following discussion outlines various approaches including: 1) ignoring only lowest ordered bits in succession; 2) ignoring the highest ordered bit and one or more successive lower ordered bits; and, 3) custom calculating which bits are to be ignored based upon the data itself. Each of these are discussed immediately below.

Ignoring Only The Lowest Ordered Bits In Succession

[0030] Ignoring the lowest order bits has the biggest impact on resolution but the smallest impact on dynamic range. **Figure 4b** demonstrates this aspect in simple terms. Before discussing **Figure 4a**, however, it is first helpful to go through the binary mathematics at play. Consider as an example frame data implemented as 8 bits per data value (i.e., N = 8); and, consider an example of ignoring only the three lowest ordered bits in succession (i.e., M = 5). In such a case, a mask of the original 8 bit value can be represented as _ _ _ _ _ X X X, ,where X is an ignored bit and _ is a non ignored bit. The full 8 bit data values are capable of expressing a dynamic range of 0 to 255 with a resolution of 1 (i.e., 256 different values: 0, 1, 2, 3, ..., 255). Ignoring the lowest ordered three bits of an 8 bit data value causes the richness of the expressible numeric values to be degraded to a dynamic range of 0 to 248 with a resolution of 8 (i.e., 31 different values: 0, 8, 16, 24, ..., 248).

[0031] **Figure 4b** attempts to graphically depict the comparison outlined above (albeit qualitatively and not to scale). The leftmost depiction 420 provides a qualitative feel for the dynamic rage and resolution at which macro block data values can be expressed when all of the bits per data value from a video frame are employed (e.g., all 8 bits as discussed in the example above). The rightmost depiction 421 provides a qualitative feel for the dynamic rage and resolution at which the same macro block data values can be expressed when a set of successive lower ordered bits per data value from a video frame are dropped (e.g., the three lowest bits are dropped as discussed in the example above).

[0032] Here, note that the data profile observed in depictions 420, 421 of **Figure 4b** can be viewed as a plot of macro block data values. For example, simply by reading macro block data values in some order (e.g., first row left to right, second row left to right, ... last row left to right) a pattern unique to the imagery of the macro block will reveal itself. The SAD circuitry essentially finds matching patterns as between the reference macro block pattern and the search window's macro block pattern. Because the space between a neighboring pair of horizontal lines corresponds to a same numeric value, higher resolution (i.e., smaller increments) is represented in the depictions of Figure 4b as less space between neighboring lines; which, in turn, corresponds to more horizontal lines superimposed over the data.

[0033] Clearly the leftmost depiction 420 has higher resolution than the rightmost depiction 421. For example, referring to the set of data points 422, note that the data points are expressed with a different numeric value in the

leftmost depiction 420 (because each data point is binned between a different pair of horizontal lines); while, for the rightmost depiction 421 these same data points must be expressed with the same data value (because they are each binned between the same pair of horizontal lines). The loss in dynamic range is depicted as a hashed region 423 where values cannot be expressed. Here, any value that under full resolution would fall in the blacklisted region 423 would be given a "clipped" maximum value at the highest space between neighboring lines.

[0034] With an understanding of the loss in dynamic range and resolution that occurs when lower ordered bits are dropped, depictions 420 and 421 help to emphasize the point that full resolution (e.g., as qualitatively observed in the leftmost depiction 420) is not necessary to identify matching patterns or at least patterns having a high degree of sameness. That is, for many practical situations, the resolution observed in depiction 421 is "good enough" to identify matching patterns. A pair of identical or nearly-identical macro block patterns as observed in depiction 421 (and expressed with the resolution and dynamic range observed in depiction 421) should correctly produce a very small SAD value (so as to mark sameness or near sameness between the pair of patterns) because the unique features of the entire pattern as a whole is weighed into the SAD calculation (e.g., the smaller left peak, the large right peak, etc.).

Ignoring The Highest Ordered Bit And One Or More Successive Lower Ordered Bits

[0035] Ignoring the highest ordered bit has a significant impact on dynamic range but allows for better resolution. As an example, consider again frame data implemented as 8 bits per data value (i.e., N = 8) where 3 three bits are to be ignored (i.e., M = 5) including the highest ordered bit and the two lowest ordered bits (i.e., a mask of the original 8 bit value can be represented as X_ _ _ _ _ X X, , where X is an ignored bit and _ is a non ignored bit). According to the numerics of such an approach, the dynamic range is more than cut in half on a linear scale; but, the resolution is better than that observed in depiction 421 of **Figure 4b**. Specifically, whereas the full 8 bit data values are capable of expressing a dynamic range of 0 to 255 with a resolution of 1 (i.e., 256 different values: 0, 1, 2, 3, ..., 255), the "X_ _ _ _ _ X X" approach is only capable of expressing a dynamic range of 0 to 124 with a resolution of 4 (i.e., 31 different values: 0, 4, 8, 12, ..., 124).

[0036] Comparing the rightmost depiction 521 of **Figure 5b** with the rightmost depiction 421 of **Figure 4b**, note that the hashed region 423, 523 that corresponds to dynamic range loss is much larger in **Figure 5b** than in **Figure 4b**; however, the resolution as measured by the space between neighboring lines is improved. The approach represented in **Figure 5b** is an ideal approach when the macro block data is as observed in **Figures 5a** and **5b**: for the most part, "stuffed" in a particular half or so of the full dynamic range. In such a situation,

because the data is "stuffed" in a particular half or so of the full dynamic range, a substantial loss in dynamic range is acceptable because the data for the most part does not exist in half of the full dynamic range space. Moreover, again because the data is stuffed into a particular region, it is helpful to improve the resolution as compared to the approach of **Figure 4b** so that the data values can be better differentiated and the overall data profile better captured as a consequence.

[0037] In order to treat data that is for the most part stuffed into a higher half of the full dynamic range (as observed in **Figures 5a** and **5b**) or a middle half of the full dynamic range, an offset should be added to the data to make it viewable within the limited dynamic range. **Figure 5a** presents a basic circuit architecture; and, depiction 520 of **Figure 5b** shows a data pattern that is stuffed into an upper half of the full dynamic range.

[0038] According to one embodiment of that the architecture of **Figure 5a** can be applied to, an offset calculation unit 505 (which can be constructed with logic circuitry) detects that the first frame's macro block data 502 manifests a pattern having data values that are stuffed into an upper half of the full dynamic range; and, in response, provides an offset of - (A-B) to effectively "drop" the data values into the viewable dynamic range as depicted in **Figure 5b** (note in **Figure 5b** that negative values after application of the offset are clipped to a value of 0). A pair of adders 504a, 504b add raw frame data values to the offset. In the embodiment of **Figure 5a** the additions are performed with N bit wide data values. The AD unit 501 (which may also be implemented with logic circuitry)

then operates on M of these N bits per data value by ignoring the highest ordered bit and one or more successive lower ordered bits.

Custom Calculating Which Bits Are To Be Ignored Based Upon The Data Itself

[0039] In another (more sophisticated) embodiment to which the architecture of **Figure 5a** can be applied, the offset calculation unit 505 is designed to not only determine an offset to be applied to the raw data but also determine which bits of the N bits per raw data value are to be masked out so that only M bits per data value are used for SAD calculation purposes. Here, a least significant bit is masked out for each of the N-M most significant bits that cannot be masked out. For example, if M=8 and N=5, then there exist four possible scenarios: 1) none of the most significant bits are masked out leaving the three least significant bits to be masked out; 2) the most significant bit is masked and the two least significant bits are masked out; 3) the two most significant bits are masked and the least significant bit is masked out; and, 4) the three most significant bits are masked out.

[0040] Thus generally, as depicted in **Figure 5c**, the number of most significant bits that can be masked out are first determined 551; and then, the number of least significant bits that can be masked out are determined 552. In one embodiment, the offset is set to the minimum data value that exists in the reference macro block worth of data and the number of most significant bits that can be masked out is the rounded down integer of

$$\log_2 \lfloor 2^M / \text{MaxValue} \rfloor$$

where MaxValue is the maximum value observed in the reference block worth of data. For example, if M=8, N = 5 and MaxValue = 128, then the number of most significant bits that can be masked out is $\log_2|2^M/\text{MaxValue}| = \log_2|256/128| = \log_2|2| = 1$ and the two least significant bits are masked out (i.e., X _ _ _ _ _ X X). As another example, if M=8, N = 5 and MaxValue = 64, then the number of most significant bits that can be masked out is $\log_2|2^M/\text{MaxValue}| = \log_2|256/64| = \log_2|4| = 2$ and the least significant bit is masked out (i.e., X X _ _ _ _ _X).

[0041] **Figure 6** shows a generic design that can be used in any system that employs video compression including the SAD techniques discussed herein. According to the system design of **Figure 6**, raw uncompressed video data is loaded into a memory 601. The memory 601 , which can be an SRAM memory or DRAM memory, is coupled to a video compression circuit 602 (e.g., having an architecture that resembles the architecture observed in **Figure 3** where memory 60 is coupled to both register 302 and RAM 303). A reference macro block and its corresponding search window is loaded into the video compression circuit 602 from the memory 601. The video compression circuit contains AD units that process less than all of the bits per raw data value (e.g., as described from **Figure 4a** through **5c**) for the purpose of performing SAD calculations.

[0042] In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without

departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.